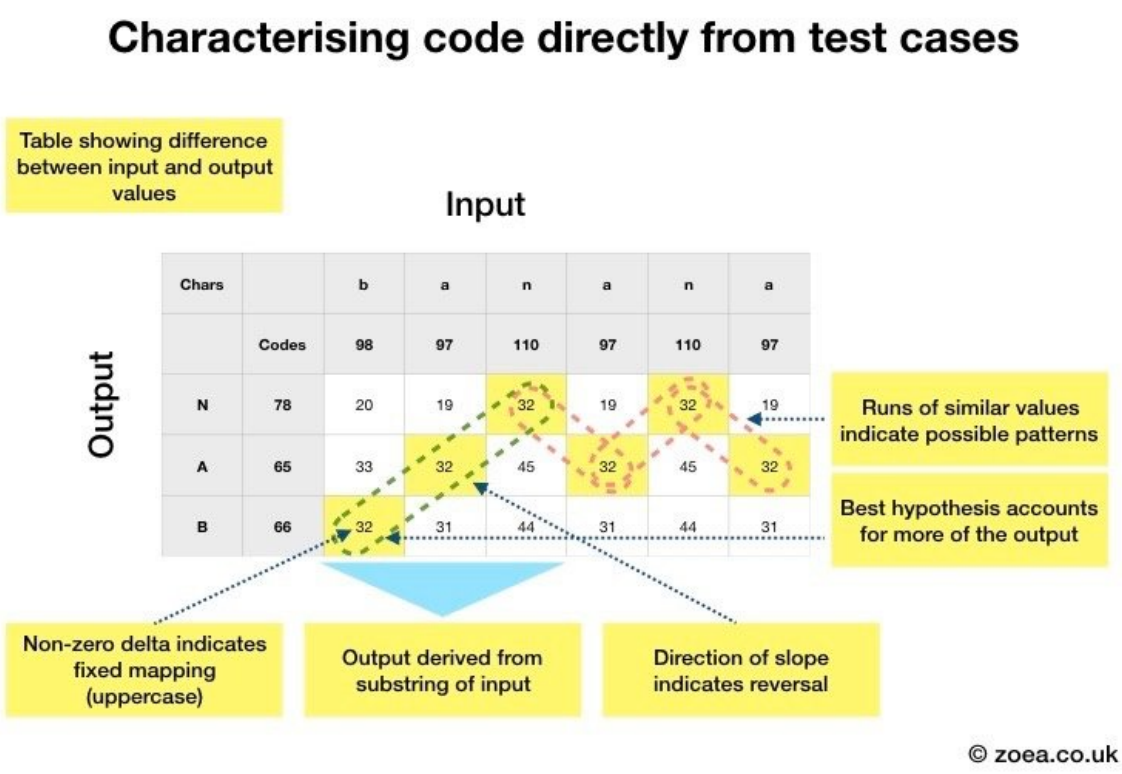
Zoea

Research Note 21

Characterising Code Directly From Test Cases

Edward McDaid & Sarah McDaid
18 Jun 2022

Computer programmers can frequently identify the corresponding code directly from one or more test cases. How can we get computers to do the same?



Zoea lets people produce software without having to learn to code. Instead users describe the program they want by providing one or more test cases. A test case is simply an

example of an input and the corresponding output. Zoea then generates the required code automatically from the test cases.

Transforming test cases into code is easier said than done. The obvious approach is simply to generate every possible program in turn and try each one to see if it matches the test cases. In theory this strategy will work (eventually) but the problem is that for non-trivial programs there are an enormous number of possibilities. This means we may have to wait days, years or even more than the lifetime of the universe for an answer.

Another approach is to examine the test case data for possible clues about the sort of processing involved. Human coders will often be given test cases as part of the specification for a new program and - depending on the type of program - they can frequently identify much of the behaviour simply by looking at the test data.

We have no real idea how people manage to accomplish this but for computers this sort of problem is called pattern recognition. The basic approach is to systematically compare the input and output data elements in different ways in order to identify any regularities that might be apparent.

Test cases can include inputs and outputs of different data types in any combination. In addition there are many different sorts of transformations that can be applied to data. As a result we employ a variety of different recognisers to detect patterns. We also use a standard set of converters to transform data between different types.

For example, if we have numeric values we can calculate the numerical difference between an input and an output. If we do this for every input against every output then we end up with a table of differences or deltas. A lot of these values are just random noise but if some of the values are the same, or similar, or even form a sequence then we have a pattern we might be interested in.

This sort of pattern recognition will often produce multiple candidate patterns and these may even overlap. As always we are interested in the smallest set of patterns that provide the simplest and most complete explanation for the available data.

The candidate patterns we identify can tell us different things about the code they correspond to. For a start the particular pattern matcher(s) involved identify the general form of the computation. In addition the scope of a candidate pattern can tell us what data elements are utilised. Depending on the recogniser additional information about the solution may also be available.

Pattern matchers in Zoea are integrated in the same way as other reasoners - as blackboard knowledge sources. Each recogniser acts on synthetic test cases and if successful produces code and further synthetic test cases. For simple problems recognisers will sometimes generate a complete solution but more often than not they provide partial solutions, hints or evidence to support a particular line of reasoning. People undoubtedly use a wide range of knowledge when thinking about code and in this respect at least Zoea is no different.

Learn more at **zoea.co.uk**